A PARALLEL PROCESSOR FOR REAL-TIME SPEECH SIGNAL PROCESSING

A.V.Ashajayanthi, S.Rajaram, N.Viswanadham

School of Automation, Indian Institute of Science
Bangalore 560012, INDIA

ABSTRACT

This paper describes the structure of a parallel processing system, primarily suitable for real-time Linear Prediction (LP) analysis and synthesis of speech signals. This system employs specially developed parallel forms of the well known algorithms (Levinson's Recursion, SIFT as well as LP synthesis filter) for this purpose. The feasibility of implementing the proposed structure using microprogrammable microprocessors, is then discussed. The proposed scheme can be readily adapted for on-line identification of autoregressive processes.

INTRODUCTION

In recent years, the mathematical technique of Linear Prediction (LP) has evolved as an efficient method for representing speech signals in terms of a small number of slowly varying parameters. This form of representation plays a key role in speech transmission, perception and automatic speech production. However, the LP analysis and synthesis procedures involve extensive computations, which make it very difficult, if not impossible, to perform them on-line, on a conventional minicomputer/microcomputer. The use of multiprocessor computer system, which offer significant speed advantages over single processor system, provide us a viable alternative for such real-time processing. The advent of low cost microprocessors, particularly the TTL, bitsliced, microprogrammable microprocessors provide all the needed speed and design flexibility while rendering such special purpose systems practical and cost effective.

The efficiency of a multiprocessing system however, depends on matching the architecture to the computational structure of the application concerned. The structural characteristics of a typical speech processing algorithms can be exploited to arrive at a relatively simple multiprocessor architecture, for meeting many functional requirements.

In this paper, we propose a multiple microprocessor architecture primarily suitable for on-line LP analysis and synthesis of speech. Further, we present parallel versions of the algorithms, for the various elements of the LP analysis and synthesis procedure, such as determination of autocorrelation function and LP coefficients, pitch extraction, and inverse filtering, which are directly applicable to the proposed structure. The paper concludes with a discussion of the suitability of this structure for other signal processing operations such as on-line identification.

SYSTEM ARCHITECTURE

The proposed system employs a modified Single Instruction Multiple Data (SIMD) structure [1], consisting of m+1 processors numbered p(0) through p(m), and interconnected as shown in fig.1. The processors p(1) through p(m) are simple arithmetic processors working in strict synchronism, as in the conventional SIMD scheme. The processor p(0), however, combines both the arithmetic function common to the other processors as well as the total system control function. The latter is achieved by including in p(0) the capability to access the instructions, decode them, and issue the necessary control signals to all the arithmetic processors for performing the indicated functions. Moreover, p(0) also serves as the controller for interfacing with external I/O devices, such as ADC and DAC's connected via its I/O Bus.

To provide maximum parallelism in operations, each of the processors is provided with its own private memory (PM), containing the data to be operated upon. Additionally, bidirectional data transfer lines, known as Interprocessor Communication Bus (IPC) connecting adjacent processors are also provided to shift data between them. Thus data can be shifted from p(i) to either p(i-1) or p(i+1) for all $i \in (0, m)$, simultaneously, with the exception that p(0) and p(m) are not considered as adjacent.

The system also incorporates a bidirectional BROADCAST Bus, so that a common data element may be transmitted from any processor to all the other processors simultaneously. Even though the same function could be achieved by successive transmissions using the basic IPC Bus, the latter approach would require m-cycles as against the single cycle needed when BCAST Bus is used.

CH1379-7/79/0000-0868$00.75

PARALLEL ALGORITHMS FOR LPC ANALYSIS AND SYNTHESIS

A primary requirement in the use of a multiprocessor scheme, such as the one described earlier in any application, is the development of suitable algorithms which can effectively utilize the multiprocessing capabilities and the available data transfer links. In the present context of LP analysis and synthesis, we require parallel forms of algorithms for (i) determination of autocorrelation functions, (ii) determination of the LP coefficients from a set of linear equations involving the autocorrelation functions, (iii) pitch extraction and (iv) inverse filtering. In this connection, we present parallel forms of algorithms suitable for performing these computations on the proposed system.

Autocorrelation LP analysis

The LP analysis of speech signals is based on the concept that the present speech sample can be predicted as a linear combination of its m predecessors. That is, if $\hat{s}(n)$ is the predicted nth sample then

$$\hat{s}(n) = -\sum_{i=1}^{m} a(i)s(n-i) \quad \ldots\ldots\ldots(1)$$

where a(i), $1 \leq i \leq m$ are the LP coefficients, s(i), for all $0 \leq i < N$ are the speech samples and N is the number of samples in a frame. The error in the prediction of nth sample is

$$e(n) = s(n) - \hat{s}(n) = \sum_{i=0}^{m} a(i)s(n-i) \quad \ldots\ldots(2)$$

Thus, to find a(i), $0 \leq i \leq m$, the total mean square error in predicting all the samples in a frame is minimized, which results in a set of m linear equations given by [2],

$$\sum_{i=1}^{m} a(i)r(n+i-1) = -r(n+1), \quad 0 \leq n \leq m \quad \ldots\ldots(3)$$

where r(j), $0 \leq j \leq m$ are the autocorrelation functions defined as

$$r(j) = \sum_{k=0}^{N-j} s(k)s(k+j), \quad j \geq 0 \quad \ldots\ldots\ldots(4)$$

The LP analysis is now performed in two stages; (i) computing the autocorrelation functions and (ii) solving the system of equations (3), by using Levinson recursion [3].

Using the system proposed we can compute all the m+1 autocorrelation functions simultaneously. The algorithm for performing this is given below employing an ALGOL like language notation, following Stone [4]. Appropriate comments have been included to make the description as self-explanatory as possible.

```
Algorithm comment Autocorrelation function
                  evaluation;
comment Initialise autocorrelation functions R to
        zero;
        R(j) := 0, (0≤j≤m);
comment The notation (0≤j≤m) following statement
        indicates that this operation is performed
simultaneously by all processors p(j), j ∈ (0, m);
comment collect the first m+1 speech samples in
        processors p(0) through p(m);
  for i:=0 step 1 until m-1 do
 begin
comment Get the new sample into processor p(0);
        P(0) : = S(i);
comment Down shift the samples;
        P(j): = P(j-1), (1≤j ≤m);
  end of i loop;
        P(0): = S(m);
comment Now the variable P(i) in the processor
 P(i) will contain speech sample S(m-i) for 0≤i≤m;
comment Evaluate autocorrelation functions;
  for j := m step 1 until N+m do
  begin
comment Broadcast P(m), the oldest of the (m+1)
        samples stored, to all the processors;
        broadcast (P(m));
comment Compute the partial sums of autocorrelation function;
        R(j):= R(j)+P(m)×P(j), (0≤j≤m);
comment Shift down the speech samples and get
        the next sample into processor p(0);
        P(j):= P(j-1), (1≤j≤m)
        P(0):= S(i)
  end of j loop;
```

In the end, any processor p(i) will contain R(m-i), for all i ∈ (0, m). Next all the autocorrelation functions are broadcast to all the processors, to enable further analysis. This will require m+1 broadcast operations.

Next, consider the Levinson's recursion procedure for computing the LP coefficients, of eqn. (3). The normal sequential method of computing a(i), i ∈ (0, m) can be described as follows using ALGOL notation for simplifying the description of the algorithm.

```
Algorithm comment Sequential procedure for
                  Levinson recursion;
comment initialization. a(m, i) denotes the value
        of LP coefficient a(i) at the end of m
        iterations;
     a(o, o) := 1;
    α(o) := R(o);
    β(o) := R(1);
 for n:=o step 1 until m-1 do
 begin
     k(n) := -β(n)/α(n);
    α(n+1) := α(n)+k(n) × β(n);
     a(n+1, o) := a(n, o);
 for i:=1 step 1 until n+1 do
 begin
     a(n+1, i) := a(n, i)+k(n)×a(n, n-i+1);
 end of i loop;
 for j:=o step 1 until n+1 do
 begin
    β(n+1) := β(n+1)+a(n+1, j)×R(n+1-j);
 end of j loop;
```

```
end of n loop;
```
a(m,i), $o \leq i \leq m$ are the required LP coefficients.

We can convert this sequential algorithm into a parallel form to take full advantage of our proposed structure, by repeating some of the computations. In fact, each processor p(i) will compute both a(n,i) and a(n,n-i), so that the basic data shifts permitted between adjacent processors, are sufficient to align the a(i) pairs appropriately for further iteration.

```
Algorithm comment parallel Levinson recursion;
  comment initialisation;
      a(o,o) := 1;
      b(o,1) := 1;
  comment in general b(i,j) will contain a(i,i-j)
      for j ∈ (o,m);
   ∝ := R(o);
   β := R(1);
  comment evaluation of a's;
  for n:=o step 1 until m-1 do
   begin
      k := -β/∝;
      ∝ := ∝ + kβ;
      broadcast (k);
      c(n+1,j) := a(n,j)+b(n,j)xk, (o≤j≤m);
      b(n+1,j) := b(n,j)+a(n,j)xk, (o≤j≤m);
      a(n+1,j) := c(n+1,j), (o≤j≤m);
        1(j) := a(n+1,j)xR(n+2-j), (o≤j≤m);
        2(j) := b(n+1,j)xR(j+1), (o≤j≤m);
  comment ⌈x⌉ denotes integral part of x;
if nmod2=1 then β(j):= β1(j)+ β2(j), (j ≠ ⌈n/2⌉+1)
  else β1(j):= β1(j)+ β2(j), (o≤j≤m);
     β := β1(o);
for j:=1 step 1 until ⌈n/2⌉+1 do
  begin comment shift up partial sums of β;
      β1(j) := β1(j+1), (o≤j≤m-1);
      comment and compute β;
        β := β + β1(o);
  end of j loop;
    b(n+1,j) := b(n+1,j-1), (1≤j≤m);
end of n loop;
```

Pitch extraction

The well known SIFT algorithm [5] for pitch extraction can be readily adapted to the suggested structure, since it essentially involves Levinson's recursion of fourth order, evaluation of residual error signals using these coefficients by inverse filtering, and evaluation of autocorrelation functions of these error signals. The computations of Levinson recursion and autocorrelation can be carried out using the parallel algorithm similar to those already given, while inverse filtering can be parallelized as described in the next sub-section. The voiced/unvoiced frame detection, however, needs to be performed sequentially.

Synthesis:

For the synthesis of speech signals, direct form realization of the all-pole synthesis filter can be converted into the parallel form of the algorithm as given below. Here e(i) indicates the excitation function which is either an impulse sequence (for voiced frames) or noise sequence (for unvoiced frames).

```
Algorithm comment synthesis;
comment initialization;
      S(0) := e(0);
      S1(j) := 0, (0≤j≤m);
for i:=1 step 1 until N-1 do
  begin
      broadcast (S(i-1));
comment Compute partial sums;
      S1(j) := S1(j)+S(i-1)xa(m,j), (1≤j≤m);
comment shift up the partial sums;
      S1(j) := S1(j+1), (0≤j≤m-1);
comment Compute speech output in processor p(0)
      S(i) := e(i)-S1(0);
end of i loop;
```

EFFECTIVE SPEED UP

While the parallel form of an algorithm exploits the availability of m+1 independent processors for performing arithmetic operations simultaneously, they involve additional overheads such as broadcasts and data shifts, for aligning the data as required. In view of these overheads, the overall speed up that results may not be m+1.

By comparing individual operations such as loading, additions, multiplications, broadcasts and data shifts in both the normal sequential form as well as the suggested parallel form, we can obtain a measure of the effective speed up possible with this system. These results are summarized in Table 1 below. In arriving at the numerical values for the effective speed up, we have assumed that the add time and load time are same, the broadcast and shift operation require twice the load time, while the multiplication operation is equivalent to 16 load/add operations (as is possible with AM2903 microprocessors).

Table I. Effective speed up for the parallel algorithms
(Computed for the case with N=256, m=13)

Procedure	Time for sequential system	Time for parallel system	Effective speed up ratio
Proc.1	55263	5667	9.75
Proc.2	-	26	-
Proc.3	2963	1446	2.04
Proc.4	53138	6121	8.68
Total	111364	13260	8.40

(Proc.1: autocorrelation evaluation, Proc.2: Broadcast autofunctions, Proc.3: Levinson recursion, Proc.4: Synthesis. All execution times shown are in terms of basic load operations)

HARDWARE REALIZATION

The availability of microprogrammable, bit-sliced microprocessors has simplified the design of array processors, such as the one described in this paper. While we have built a processor with very similar architecture, on an experimental basis earlier, using INTEL-3000 family of devices [6], the availability of more powerful bit-slices as the AM2900 family of devices, is found to be more attractive for building a complete processor for such real-time speech applications.

Specifically, each of the arithmetic processors p(1) through p(m), is to be a 16 bit unit, realised using four numbers of AM2903 microprocessor slices, along with a AM2902 carry look ahead generator. The processor p(0), will in addition, contain three AM2909 microsequencers cascaded together, which together with a ROM containing microinstructions will function as the central microprogrammed control element. It is estimated that about 512 words of 60 bits each will be required as the control memory. The usual commercially available semiconductor RAM's with an access time 300 nsecs are proposed to be used as the private memories.

SOME POSSIBLE EXTENSIONS

The algorithms presented above can be extended to include more versatile speech transmission schemes such as the adaptive frame rate and improved modelling of speech using overlapped frame analysis. In transmission schemes using adaptive frame rates, the LP coefficients are transmitted only when the distance function determined, based on the autocorrelation functions of the speech samples of the current frame and those of the LP coefficients of the previous frame, exceeds certain threshold value[7],[8]. The distance function can be computed in paralle, in a similar manner as described earlier. In the case of the overlapped frame analysis, the computation requirement is one of calculating two sets of autocorrelation functions - one for the normal succession of speech frame and the other for the overlapped frame, and also the respective LP coefficients. The algorithm already described could be modified to suit the above requirement.

Also, the Levinson's recursion algorithm described above (where the order of recursion is known a priori), can be modified to suit the requirement for on-line identification of AR process [9], (which requires that the recursion be continued till the difference in residual error between successive iterations becomes constant).

CONCLUSION

We have described the structure of a parallel processing system, employing microprogrammable microprocessors, which can be adapted for a number of real-time signal processing tasks such as LP analysis and synthesis of speech, and on-line identification of autoregressive process. The parallel forms of basic algorithms needed for use with this structure are also described along with a quantitative estimate of the speed benefits that would result. A specific hardware realisation of this unit, employing the AM2900 family of microprocessors is in progress.

References

[1] M.J. Flynn, "Very high speed Computing systems," Proc. IEEE Vol. 54, pp. 1901-09: Dec. 1966.

[2] J.D. Markel and A.H. Gray, Jr. Linear prediction of speech, New York: Springer-verlag 1976.

[3] N. Levinson, "The Weiner RMS error criterion in filter design and prediction," J. Math. Phys. Vol. 25, pp. 261-278; 1974.

[4] H.S. Stone, Introduction to Computer Architecture, Chicago: SRA, 1975.

[5] J.D. Markel, "The SIFT algorithm for fundamental frequency estimation," IEEE Acoustics. Vol. AU-20, pp. 367-377: Dec. 1972.

[6] V.K. Prasanna Kumar, H.K. Sampath Kumar, "Vector instruction processor VIP-3000," ME project report, School of Automation, Indian Institute of Science, Bangalore, 1978.

[7] D.J. Magill, "Adaptive speech compression for Packet Communication Systems," Telecom. conf. Record, IEEE Pub. 73CHO805-2, 29D1-5, 1973.

[8] F. Itakura, "Minimum prediction Residual Principle Applied to Speech Recognition", IEEE ASSP, Vol. ASSP-23, pp. 67-72: Feb. 1975.

[9] M.D. Srinath, M.M. Viswanathan, "Sequential algorithm for identification of parameters of autoregressive process," IEEE Aut. Control, Vol. AC-20, pp. 542-546: Aug. 1975.

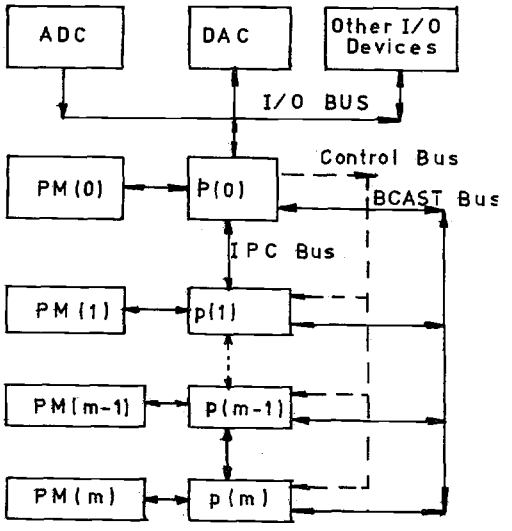

FIG.1. PROPOSED STRUCTURE